Заключительный этап

Заключительный этап олимпиады состоит из двух частей: индивидуальное решение задач по предметам (математика, информатика) и командное решение задачи на создание прототипа программного обеспечения.

Индивидульный предметный тур

На индивидуальное решение задач дается по 2 часа на один предмет. Для каждой из параллелей (9 класс или 10-11 класс) предлагается свой набор задач по математике, задачи по информатике - общие для всех участников.

Решение каждой задачи по математике дает определенное количество баллов (см. критерии оценки). При этом некоторые задачи делятся на подзадачи. За каждую подзадачу можно получить от 0 до указанного количества баллов.

Решение задач по информатике предполагало написание программ. Ограничения по используемым языкам программирования не было. Проверочные тесты для каждой задачи по информатике делились на несколько групп. Прохождение всех тестов в группе тестов дает определенное количество баллов за решение задачи.

Участники получают оценку за решение задач в совокупности по всем предметам данного профиля (математика и информатика) — суммарно от 0 до 200 баллов:

- **Математика 9 класс** количество набранных баллов (от 0 до 100);
- **Математика 10-11 класс** количество набранных баллов (от 0 до 100);
- **Информатика** количество набранных баллов (от 0 до 300) делится на коэффициент 3.

Информатика. 8-11 класс

Задача III.1.3.1. Комната ярости 2 (100 баллов)

Нашего героя зовут Коля, и мы застали его не в лучшем расположении духа. Благо его добрая подруга Гертруда была неподалёку и убедила Колю пойти выпустить пар в ближайшую “Комнату ярости”.

Со слухом у Николая всё в порядке, поэтому предметам в “Комнате” пощады ждать не придётся. Вернее, в “Комнатах”, ведь новые времена требуют новых решений и “Комната ярости” разрослась практически до “тоннеля”, по которому Николай собирается пронестись мощнейшим ураганом.

В ближайшей “Комнате ярости” n комнат, которые последовательно соединены коридорами. То есть первая комната, с которой и начнёт Николай, соединена коридором только со второй комнатой, i-я ($1 < i < n$) комната соединена с $i - 1$-й и $i + 1$-й, а n-я соединена только с $n - 1$-й комнатой. Каждый из коридоров Николай преодолевает за единицу времени, абсолютно не тратя на это сил. Также в каждой комнате есть дверь, которая позволяет покинуть “Тоннель ярости”, но наш герой не планирует выходить из “Тоннеля” ни из какой комнаты, кроме n-й, а из n-й выйдет только после того, как уничтожит всё, что можно уничтожить во всех n комнатах.

Единственное, что может остановить Николая в те моменты, когда хочется крушить всё, что попадается на глаза, – это усталость. Изначально у Коли есть e единиц энергии и ни в какой момент времени не может быть больше e единиц энергии. А при нуле энергии Коля теряет способность крушить предметы и максимум может ходить по комнатам и коридорам, как призрак, пока у него не появятся новые силы.

Новые времена – это ещё и новые технологии. Вот и в “Комнатах ярости” появились роботизированные мини-бары по одному на комнату. В них можно найти практически всё, что душе угодно.

Да, силы Николая не безграничны. Но он здоров как бык. И потому он давно для себя выбрал самый доступный, дешёвый и крайне вредный способ пополнения сил – заправляться энергетиками.

В мини-барах есть энергетики, но всего лишь одной марки. Изначально в мини-баре i-й комнаты есть k_i банок энергетика, каждая из которых может прибавить одну единицу энергии Коле.

После каждого подхода Николая к мини-бару соответствующий мини-бар обновляет себя: убирает все банки энергетика, что в нём были, и достаёт новые. В общем случае, при j-м ($j \geqslant 1$) подходе Николая к мини-бару в i-й комнате в мини-баре будет $\max(0, k_i + 1 - j)$ банок энергетика, каждая из которых может прибавить j единиц энергии Николаю.

Николай уже выбрал стратегию, которой он будет следовать в “Тоннеле”. Он будет разрушать предметы последовательно: сначала все предметы в первой комнате,

потом все предметы во второй комнате и т. д., пока не уничтожит всё. Переносить предметы из одной комнаты в другую он не будет. На уничтожение одного предмета он тратит единицу своей энергии и единицу времени. Делать паузы он будет только в моменты, когда его энергия иссякнет, а в комнате, где он находится, ещё не все предметы разрушены.

В такой ситуации Николай сначала пойдёт к мини-бару той комнаты, в которой он сейчас находится, и выпьет все энергетики из этого мини-бара, даже если некоторые из них не прибавят ему энергии. Более формально, если текущий запас энергии Коли равен x единиц энергии и он собирается выпить банку энергетика, которая может прибавить y единиц энергии, то после выпивания этой банки запас энергии Коли будет равен $\min(x + y, e)$ единиц энергии.

Если энергия по-прежнему на нуле, то он выбирает одну из комнат, в которых он уже разрушил все предметы, и выпивает все энергетики из мини-бара этой комнаты. Из всех таких комнат он выбирает ту, энергетики в мини-баре которой прибавят Николаю больше всего энергии. Если и таких комнат несколько – ближайшую из них.

Для последнего описанного действия Николаю приходится напрячь мозги, чего ему совсем не хочется и еле-еле можется. Если Николаю придётся включать голову более 1000 (тысячи) раз за время нахождения в “Комнате”, то на 1001-й он потеряет ориентацию в пространстве и не сможет довести дело до конца.

Если после опустошения одного из мини-баров энергия Николая оказывается больше нуля, то он сразу идёт продолжать крушить предметы.

На перемещение внутри комнаты Николай не тратит ни времени, ни сил, а на выпивание одной банки энергетика тратит только единицу времени.

Давайте рассчитаем, на какое время Гертруда точно изолировала нашего героя от общества.

Формат входных данных

В первой строке даны два целых числа n ($1 \leqslant n \leqslant 100\,000$) и e ($0 \leqslant e \leqslant 10^{18}$) – количество комнат в “Комнате ярости” и максимальный размер запаса энергии Коли.

Во второй строке n целых чисел a_i ($0 \leqslant a_i \leqslant 10^{18}$) – изначальное количество предметов в i-й комнате.

Гарантируется, что суммарное количество предметов в комнатах не превышает 10^{18}.

В третьей строке n целых чисел k_i ($0 \leqslant k_i \leqslant 10^6$) – изначальное количество энергетиков в мини-баре i-й комнаты.

Формат выходных данных

Выведите -1, если наш герой не выберется из комнат самостоятельно. В противном случае выведите минимальное количество единиц времени, которое Николай проведёт в “Комнате ярости”.

Примеры

Пример №1

Стандартный ввод
3 2 0 0 3 1 1 0
Стандартный вывод
8

Пояснение к примеру

В первом тесте Коля спокойно пройдёт до последней комнаты за 2-е единицы времени, потом разрушит два предмета за 2-е единицы времени и пойдёт во вторую комнату, чтобы выпить энергетик, вернуться, уничтожить последний предмет и уйти. Таким образом, ответ – это $2 + 2 + 1 + 1 + 1 + 1 = 8$.

Пример №2

Стандартный ввод
5 5 1 2 3 4 5 3 2 3 2 2
Стандартный вывод
33

Способ оценки работы

Подзадача	Баллы	Дополнительные ограничения	Необходимые подзадачи
0	0	Тесты из условия	–
1	10	$k_i \leqslant 1$	–
2	10	$n \leqslant 1000$, $k_i \leqslant 100$	0
3	15	$k_i \leqslant 100$	0, 1, 2
4	15	$n \leqslant 1000$	0, 2
5	20	$k_i^2 \leqslant e$	–
6	15	Николаю не придётся включать мозг	–
7	15	Нет дополнительных ограничений	0, 1, 2, 3, 4, 5, 6

Баллы за подзадачу будут начислены в случае прохождения всех тестов необходимых подзадач и всех тестов самой подзадачи.

Для проверки результата использовался следующий код на языке C++

```cpp
#include "testlib.h"
#include <sstream>

using namespace std;

int main(int argc, char * argv[]) {
  setName("compare ordered sequences of signed int%d numbers",
    8 * int(sizeof(long long)));

  registerTestlibCmd(argc, argv);

  int n = 0;
  string firstElems;

  while (!ans.seekEof() && !ouf.seekEof()) {
    n++;
    long long j = ans.readLong();
    long long p = ouf.readLong();
    if (j != p)
      quitf(_wa, "%d%s numbers differ - expected: '%s', found: '%s'", n,
        englishEnding(n).c_str(), vtos(j).c_str(),
        vtos(p).c_str());
    else if (n <= 5) {
      if (firstElems.length() > 0)
        firstElems += " ";
      firstElems += vtos(j);
    }
  }

  int extraInAnsCount = 0;

  while (!ans.seekEof()) {
    ans.readLong();
    extraInAnsCount++;
  }

  int extraInOufCount = 0;

  while (!ouf.seekEof()) {
    ouf.readLong();
    extraInOufCount++;
  }

  if (extraInAnsCount > 0)
    quitf(_wa,
      "Answer contains longer sequence [length = %d], but output contains %d
      ↪  elements",
      n + extraInAnsCount, n);

  if (extraInOufCount > 0)
    quitf(_wa,
      "Output contains longer sequence [length = %d], but answer contains %d
      ↪  elements",
      n + extraInOufCount, n);

  if (n <= 5)
    quitf(_ok, "%d number(s): \"%s\"", n, compress(firstElems).c_str());
  else
    quitf(_ok, "%d numbers", n);
}
```

Решение

Данная задача – задача на моделирование с дальнейшей оптимизацией. Для начала важно внимательно прочесть условие и правильно смоделировать описанные события. Так получается решение с асимптотикой $O\Big(\sum\limits_{i=1}^{n}(k_i) + b \cdot n\Big)$, где $b = 1000$.

При $k_i \leqslant 1$ в мини-баре каждой комнаты либо есть одна банка энергетика, либо нет ни одной и никогда больше не появится. Поэтому для быстрого определения комнаты в случаях, когда Николаю приходится включать мозг, выгодно воспользоваться стеком.

При $k_i > 1$ стек уже не подойдёт. Нам нужна структура данных, которая умеет быстро добавлять в себя элемент и удалять из себя «лучший» элемент. Для этого подойдёт, например, очередь с приоритетом.

Для полного решения остаётся лишь научиться быстро определять, сколько раз Николай подойдёт к мини-бару в i-й комнате, когда будет разрушать предметы i-й комнаты.

За j-й подход к мини-бару i-й комнаты Николай может получить

$$j \cdot (k_i + 1 - j) = j \cdot k_i - j \cdot (j - 1)$$

единиц энергии.

Так как достаточно рассматривать лишь такие j, что $1 \leqslant j \leqslant k_i \leqslant 10^6$, значения $\sum\limits_{m=0}^{j}(m)$ и $\sum\limits_{m=0}^{j}(m \cdot (m-1))$ можно предподсчитать.

Пусть

$$sum_j = \sum_{m=0}^{j}(m) = \tfrac{(1+j)\cdot j}{2}\ ,$$

$$diff_j = \sum_{m=0}^{j}(m \cdot (m-1)) = \begin{cases} 0, & \text{если } j = 0; \\ diff_{j-1} + j \cdot (j-1), & \text{если } 1 \leqslant j \leqslant k_i; \end{cases}.$$

Тогда за первые j $(1 \leqslant j \leqslant k_i)$ подходов к мини-бару в i-й комнате Николай получит не более $sum_j \cdot k_i - diff_j$ единиц энергии и потратит на эти подходы ровно $\frac{(k_i \cdot 2 + 1 - j)\cdot j}{2}$ единиц времени. Сколько именно энергии получит Николай за первые j $(1 \leqslant j \leqslant k_i)$ подходов, зависит от e.

Не сложно заметить, что график функции $f(j) = j \cdot (k_i + 1 - j)$ при фиксированном k_i является параболой с ветвями вниз. Следовательно, существует конкретное наибольшее количество энергии, которое может получить Николай за один подход к мини-бару в i-й комнате. Так как мы работаем в целых числах, это значение достигается при $j = \left\lfloor \frac{k_i+1}{2} \right\rfloor$ и равно $\max_{k_i} = \left\lfloor \frac{k_i+1}{2} \right\rfloor \cdot \left(k_i + 1 - \left\lfloor \frac{k_i+1}{2} \right\rfloor\right) = \left\lfloor \frac{k_i+1}{2} \right\rfloor \cdot \left\lfloor \frac{k_i+2}{2} \right\rfloor$.

Если $\max_{k_i} \leqslant e$, то количество раз, которое Николай подойдёт к мини-бару в i-й комнате, когда будет разрушать предметы i-й комнаты, можно найти с помощью

бинарного поиска, так как за первые j подходов Николай получит в сумме ровно $sum_j \cdot k_i - diff_j$ единиц энергии (разумеется с учётом того, что менее чем за j подходов Николай не уничтожит все предметы в i-й комнате и $1 \leqslant j \leqslant k_i$).

Если $\max_{k_i} > e$, то бинарный поиск также можно применить, но с дополнительной модификацией.

Пусть s_j – это суммарное количество энергии, которое получит Николай за первые j подходов к мини-бару в i-й комнате. Тогда:

$$s_j = \begin{cases} sum_j \cdot k_i - diff_j, & \text{если } j \leqslant c_i; \\ sum_{c_i} \cdot k_i - diff_{c_i} + e \cdot (j - c_i), & \text{если } c_i < j \leqslant k_i - c_i; \\ (sum_{c_i} \cdot k_i - diff_{c_i}) \cdot 2 + e \cdot (k_i - 2 \cdot c_i) - (sum_{k_i - j} \cdot k_i - diff_{k_i - j}), & \text{если } k_i - c_i < j \leqslant k_i. \end{cases}$$

Значение c_i для фиксированных k_i и e можно определить, например, также бинарным поиском.

Пример программы-решения

Ниже представлено решение на языке Python 3

```python
from heapq import heappop, heappush

# считываем данные
n, e = [int(i) for i in input().split()]
a = [int(i) for i in input().split()]
k = [int(i) for i in input().split()]
j = [1] * n

# текущее время, энергия и количество включений мозга
t, cur_e, brain_on_cnt = -1, e, 0
# очередь с приоритетом
heap = []

# предподсчёт
max_k = max(k)
sum_k = [0] * (max_k + 1)
diff = [0] * (max_k + 1)
for i in range(1, max_k + 1):
    sum_k[i] = sum_k[i - 1] + i
    diff[i] = diff[i - 1] + i * (i - 1)

for i in range(n):
    # входим в i-ю комнату
    t += 1

    # уничтожаем все предметы, что можем
    m = min(cur_e, a[i])
    t += m
    cur_e -= m
    a[i] -= m

    # заходим в мини-бар i-й комнаты столько раз, сколько нужно
    if a[i] > 0:
        # если энергия от энергетиков «не пропадёт»
```

```
        if (k[i] // 2 + 1) * ((k[i] + 1) // 2) <= e:
            # определяем, сколько раз Николай подойдёт к мини-бару в текущей комнате
            l, r = 0, k[i] + 1
            while l + 1 < r:
                m = (l + r) // 2
                if sum_k[m] * k[i] - diff[m] < a[i]:
                    l = m
                else:
                    r = m
            r = min(k[i], r)  # ровно r раз

            t += (k[i] * 2 - r + 1) * r // 2
            cur_e = sum_k[r] * k[i] - diff[r]
            j[i] = r + 1

            # снова уничтожаем всё, что можем
            m = min(cur_e, a[i])
            t += m
            cur_e -= m
            a[i] -= m

        # если энергия от энергетиков «пропадёт»
        else:
            # определяем, сколько первых подходов произойдут без потери энергии
            l, r = 0, (k[i] + 1) // 2
            while l + 1 < r:
                m = (l + r) // 2
                if m * (k[i] + 1 - m) > e:
                    r = m
                else:
                    l = m
            cl = l  # ровно cl подходов

            # определяем, сколько раз Николай подойдёт к мини-бару в текущей комнате
            l, r = 0, k[i] + 1
            while l + 1 < r:
                m = (l + r) // 2

                if m <= cl:
                    cur_sum = sum_k[m] * k[i] - diff[m]
                elif m <= cl + k[i] - 2 * cl:
                    cur_sum = sum_k[cl] * k[i] - diff[cl] + e * (m - cl)
                else:
                    cur_sum = (sum_k[cl] * k[i] - diff[cl]) * 2 + e * (k[i] - 2 * cl)
                    ↪ \
                              - sum_k[k[i] - m] * k[i] + diff[k[i] - m]

                if cur_sum < a[i]:
                    l = m
                else:
                    r = m
            r = min(k[i], r)  # ровно r раз

            t += (k[i] * 2 - r + 1) * r // 2
            if r <= cl:
                cur_e = sum_k[r] * k[i] - diff[r]
            elif r <= cl + k[i] - 2 * cl:
                cur_e = sum_k[cl] * k[i] - diff[cl] + e * (r - cl)
            else:
                cur_e = (sum_k[cl] * k[i] - diff[cl]) * 2 + e * (k[i] - 2 * cl) \
```

```
                        - sum_k[k[i] - r] * k[i] + diff[k[i] - r]
            j[i] = r + 1

            # снова уничтожаем всё, что можем
            m = min(cur_e, a[i])
            t += m
            cur_e -= m
            a[i] -= m

    # идём в другие комнаты за энергией, если можно и нужно
    while a[i] > 0 and brain_on_cnt < 1000 and len(heap) > 0:
        brain_on_cnt += 1
        cur_e = -heap[0][0]
        id = -heappop(heap)[1]
        t += (i - id) * 2 + k[id] + 1 - j[id]
        j[id] += 1
        if j[id] <= k[id]:
            heappush(heap, [-min(e, j[id] * (k[id] + 1 - j[id])), -id])

        # снова уничтожаем всё, что можем
        m = min(cur_e, a[i])
        t += m
        cur_e -= m
        a[i] -= m


    # если в комнате остались предметы
    if a[i] > 0:
        print(-1)  # то Николай потерялся
        exit(0)

    # иначе осталось добавить в очередь с приоритетом
    # информацию о комнате, если в ней есть энергетики
    if j[i] <= k[i]:
        heappush(heap, [-min(e, j[i] * (k[i] + 1 - j[i])), -i])

# и вывод ответа в конце программы
print(t)
```

Задача III.1.3.2. Дрон Ло (100 баллов)

Это интерактивная задача.

После “Туннеля ярости” наш Николай вернулся в реальную жизнь и приехал домой к своей Вике – вот кто по-настоящему развлекался всё это время. Оказывается, недалеко от нас, на соседних планетах живут Зентради и Атария. И Вика помогает Зентради доставить подарок Атарии.

– Что, кто, кому, как? Стоп. Ничего не понимаю. Давай с начала.

– С начала. Вчера Зентради от чистого сердца подарил мне инопланетного дрона. Причин он не назвал – кто этих инопланетян разберёт.

– Таак.

– Описание у дрона непонятное. Ну, на инопланетном языке. Но я потыкалась и немного поняла, как он работает. Я даже дала дрону имя – я зову его Дрон Ло.

– Прелестно.

– Подожди. Сегодня Зентради позвонил и попросил помочь ему. Он забыл в Дрон Ло подарок для Атарии и просит, чтобы Дрон Ло сбросил его в определённую точку планеты Атарии. Сказал, что Дрон Ло знает координаты. Я, конечно же, согласилась и ещё раз поблагодарила за замечательный подарок. Но с функциями полёта я совсем запуталась. Пыталась перезвонить Зентради, но он не берёт трубку.

И на данный момент без инопланетян, которые куда-то подевались и не отвечают на звонки, Вика мало что может понять. Но пару функций она освоила:

- Можно сделать запрос о любой точке на планете Атарии. В качестве ответа на запрос придёт расстояние от этой точки до точки, куда нужно сбросить подарок;
- Можно отправить Дрон Ло сбросить подарок в заданную нами точку.

Все точки планеты Атарии проверять не имеет смысла, так как Вика определила регион, в который нужно сбрасывать подарок и этот регион можно представить в виде двумерной плоскости.

Помогите Вике определить, куда нужно отправить Дрон Ло, и отправить его. Постарайтесь при этом сделать не очень много запросов.

Протокол взаимодействия

В данной задаче вам предстоит работать не со стандартным вводом-выводом, а со специальной программой – интерактором. Взаимодействие с ней осуществляется через стандартные потоки ввода-вывода.

Каждый запрос вашей программы должен состоять из одной строки следующего вида: `? x y` , где x и y ($-10^{18} \leqslant x, y \leqslant 10^{18}$) – дробные числа, обозначающие абсциссу и ординату точки, ответ на запрос о которой вы хотите узнать. Ответом на запрос будет расстояние от этой точки до точки, куда нужно сбросить подарок.

Когда вы решите отправить Дрон Ло сбросить подарок, выведите `! x y`, где x и y ($-10^{18} \leqslant x, y \leqslant 10^{18}$) – дробные числа, обозначающие абсциссу и ординату точки, куда вы решили отправить Дрон Ло. После этого вывода ваша программа должна завершиться и ничего больше не выводить в стандартный поток вывода.

Подарок будет считаться доставленным, если расстояние между точкой, куда вы отправили Дрон Ло, и точкой, куда отправить его было нужно, не превышает 1 (единицы).

Убедитесь, что вы выводите символ перевода строки и очищаете буфер потока вывода (команда `flush` языка) после каждой выведенной команды. На C++ для printf надо использовать функцию `fflush(stdout)`, для cout – `cout « flush` или `cout.flush()`; на Java для System.out вызов `System.out.flush()`, для PrintWriter pw – `pw.flush()`; на Pascal – `flush(output)`; на Python для sys.stdout – sys.stdout.flush(), для print – print(..., flush=True).

Примеры

Пример №1

Стандартный ввод
1.812474491323 1.234868311003 0.4356714247987 0.355991865445 0.1204635151743
Стандартный вывод
? 0.53264 0.48924 ? 0.87407 1.27174 ? 1.86586 1.02896 ? 1.77805 1.54036 ? 2.20679 1.33373 ! 2.20679 1.33373

Пояснения к примеру

В примере подарок нужно было сбросить в точку $(2.1, 1.39)$.

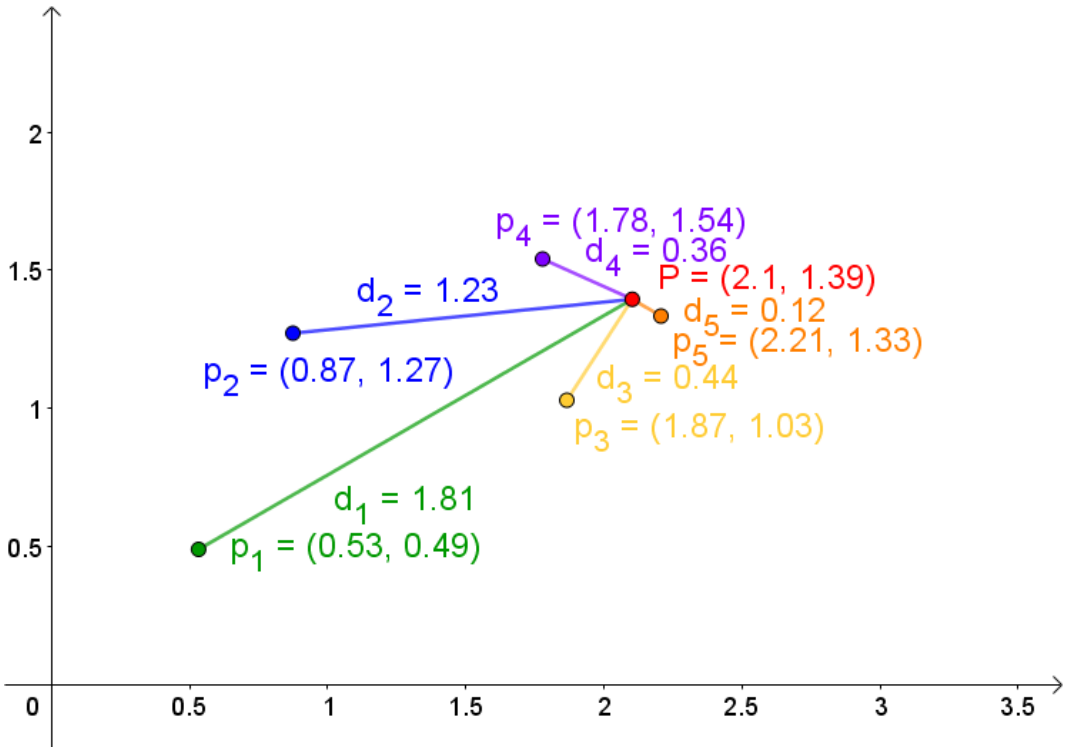


Рис. III.1.3: Рисунок к примеру 1

Способ оценки работы

В таблице ниже k – это максимальное количество запросов, которое вы можете сделать в тесте, R – расстояние от точки с координатами $(0, 0)$ до точки, куда нужно сбросить подарок, x и y – координаты точки, куда требуется сбросить подарок.

Если решение совершит более k запросов, то решение получит вердикт `Wrong Answer`.

Подзадача	Баллы	Дополнительные ограничения	Необходимые подзадачи
0	0	Тест из условия; $k = 100\,000$	–
1	20	$R \leqslant 100, k = 100\,000$	0
2	20	$R \leqslant 1000, k = 10\,000$	0, 1
3	15	$R \leqslant 10\,000, k = 1000$	0, 1, 2
4	15	$R \leqslant 10^6, k = 500$	0, 1, 2, 3
5	10	$\lvert x\rvert, \lvert y\rvert \leqslant 10^8, k = 100$	0, 1, 2, 3, 4
6	20	$\lvert x\rvert, \lvert y\rvert \leqslant 10^9, m \leqslant k \leqslant 100$	0, 1, 2, 3, 4, 5

Баллы за подзадачу будут начислены в случае прохождения всех тестов необходимых подзадач и всех тестов самой подзадачи.

Но поскольку и тут не обошлось без НЛО и теперь только инопланетяне знают число m, в последней подзадаче объявляется **потестовая оценка**. То есть баллы за последнюю подзадачу (с учётом того, что решение прошло все остальные подзадачи) будут начисляться пропорционально количеству пройденных тестов.

Для проверки результата использовался следующий код на языке C++

```
#include <bits/stdc++.h>

#include "testlib.h"

#
define forn(i, n) for (int i = 0; i < int(n); i++)

  using namespace std;

long double
dist(long double x1, long double y1, long double x2, long double y2) {
  return sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2));
}

int
main(int argc, char * argv[]) {
  setName("Interactor for 'Collective password' problem.");
  registerInteraction(argc, argv);

  long double x = inf.readDouble(), y = inf.readDouble();
  int max_queries = inf.readInt();

  int queries = 0;
  while (true) {
    queries++;

    string q = ouf.readToken();

    if (q != "!" && q != "?")
      quitf(_wa, "String '%s' from stdin is incorrect", q.c_str());
```

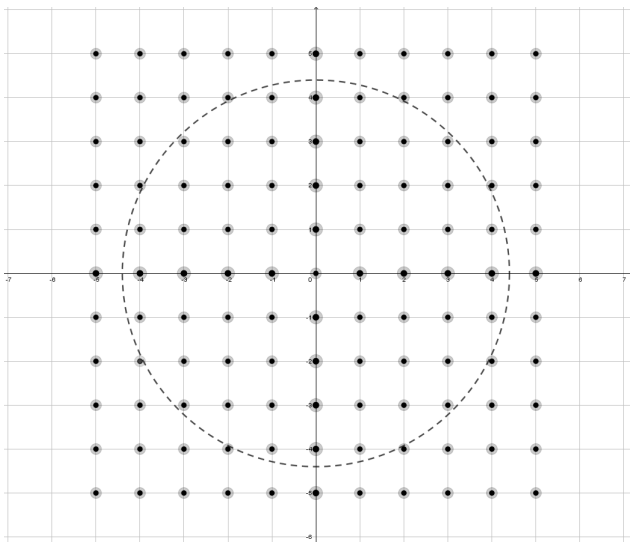
```
31
32      if (q == "?" && queries > max_queries)
33        quitf(_wa, "So mach queries");
34
35      if (q == "!") {
36        long double px = ouf.readDouble(-1e18, 1e18, "px"), py =
37          ouf.readDouble(-1e18, 1e18, "py");
38        long double pd = dist(x, y, px, py);
39
40        if (pd > 1.0)
41          quitf(_wa, "Wrong guest [%Lf ? %Lf, %Lf ? %Lf, %Lf]", px, x, py,
42            y, pd);
43
44        tout << fixed << setprecision(20) << queries << " " << px << " " <<
45          py << " " << pd << endl;
46        quitf(_ok, "%Lf %Lf is guessed (%Lf %Lf %Lf)", x, y, px, py, pd);
47      }
48
49      long double px = ouf.readDouble(-1e18, 1e18, "px"), py =
50        ouf.readDouble(-1e18, 1e18, "py");
51      cout << fixed << setprecision(20) << dist(x, y, px,
52        py) << endl << flush;
53    }
54  }
```

Решение

Подзадача 1

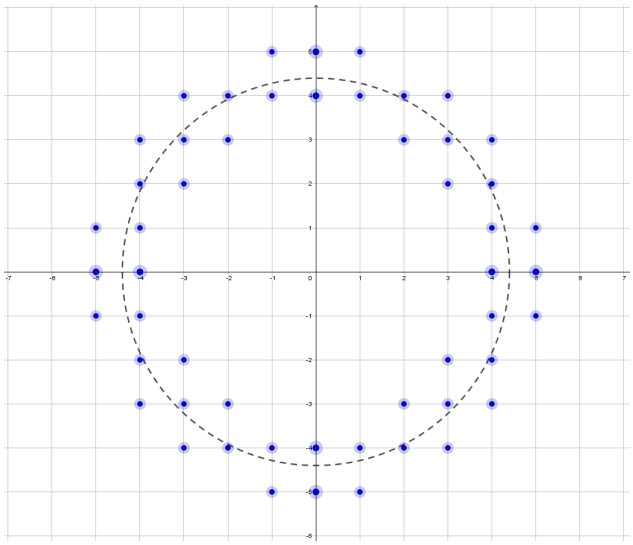
Заметим, что так как сбросить подарок можно в любую точку, расстояние от которой до загаданной не превышает 1, то существует подходящая точка с целочисленными координатами.

«Пробьём» во все такие точки следующего вида:

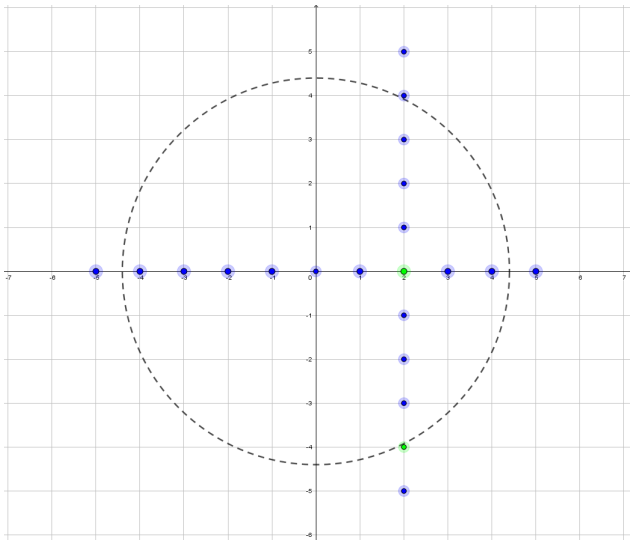


Подзадача 2

Заметим, что достаточно запрашивать про точки, которые находятся близко к окружности.

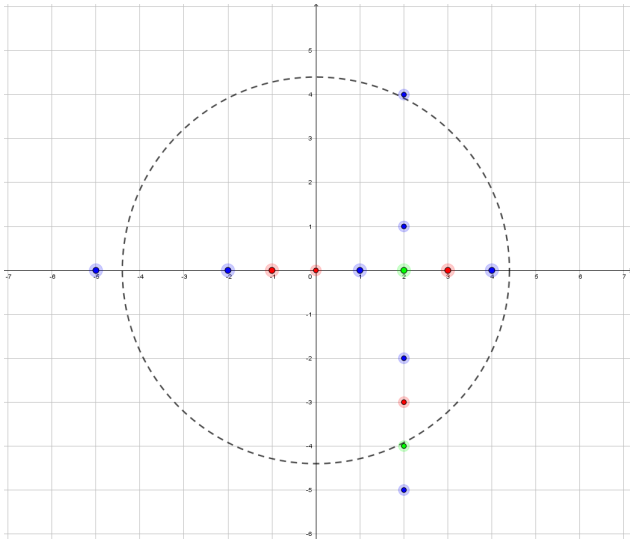


Либо можно найти ближайшую точку на оси X, а потом «пройтись» параллельно оси Y.



Подзадача 3

То же самое можно сделать перебором с шагом.



Подзадачи 4 и 5

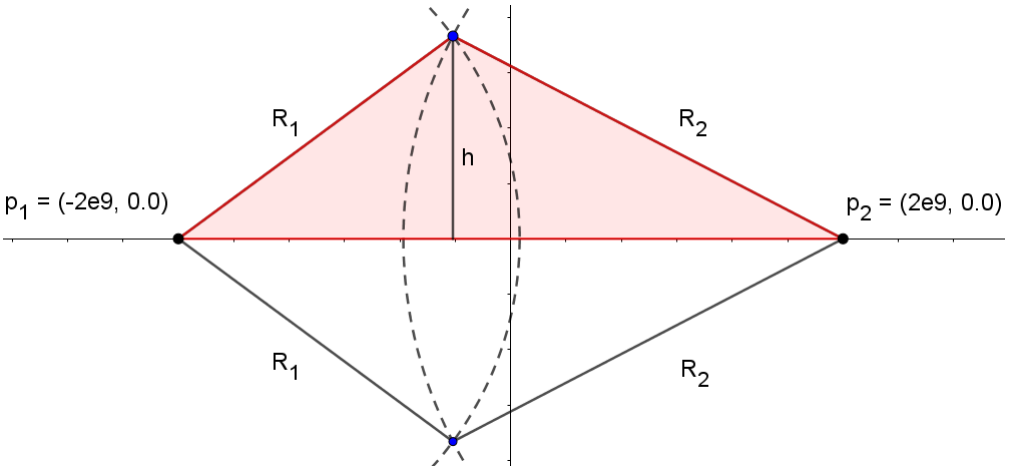
Заметим, что:

1. Найти ближайшую точку на оси X можно тернарным поиском.
2. Расстояние от ближайшей точки на оси X до загаданной точки примерно равно модулю координаты загаданной точки по оси Y
3. Чтобы не запрашивать о некоторых точках несколько раз, можно, например, сохранять результаты запросов в словарь

Полное решение

Можно сделать два запроса и найти точки пересечения двух окружностей (таким образом, $m = 3$).

Для этого можно, например, воспользоваться формулой Герона. Правда при такой версии решения требуется точность вычислений больше, чем могут предложить стандартные типы данных для работы с дробными числами.



$p = \frac{R_1+R_2+4\cdot 10^9}{2}$

$\frac{h\cdot 4\cdot 10^9}{2} = h \cdot 2 \cdot 10^9 = \sqrt{p \cdot (p-R_1) \cdot (p-R_2) \cdot (p-4\cdot 10^9)}$

$|y| = h = \sqrt{p \cdot (p-R_1) \cdot (p-R_2) \cdot (p-4\cdot 10^9)}/(2 \cdot 10^9)$

$x = \sqrt{R_1^2 - h^2} - 2 \cdot 10^9$

Пример программы-решения

Ниже представлено решение на языке Python 3

```
from decimal import Decimal

maxDist = Decimal(2e9)
print('?', -maxDist, 0, flush=True)
distLeft = Decimal(input())

print('?', maxDist, 0, flush=True)
distRight = Decimal(input())

p = (distLeft + distRight + Decimal(2.0) * maxDist) / Decimal(2.0)
S = (p * (p - distLeft) * (p - distRight) * (p - Decimal(2.0) * maxDist)).sqrt()
```

```
h = S / maxDist
x = (distLeft * distLeft - h * h).sqrt() - maxDist

print('?', x, h, flush=True)
dist = Decimal(input())

print('!', x, (h if dist < Decimal(1.0) else -h), flush=True)
```

Задача III.1.3.3. Нестандартное лечение (100 баллов)

Никто не хочет стать пациентом психиатрических больниц. Но иногда получается так, что другого выхода нет – человека нужно лечить. Психиатрическая больница имени Гаценко – это пока единственное место, где применяют новый метод с нейровирусом. Собственно, на досуге врачи этой больницы и изобрели этот полезный вирус и метод его применения.

Особенность этого нейровируса в том, что он временно "поглощает" память. При введении лучшего штамма этого нейровируса вся имеющаяся память больного сводится к отдельным фрагментам воспоминаний, имеющим ассоциации с другими фрагментами. Информация о каждом из фрагментов оказывается в отдельном нейроне мозга (то есть один такой нейрон хранит информацию только об одном фрагменте), а существование ассоциации между двумя фрагментами обеспечивают связи, своего рода мосты, между соответствующими нейронами. Главное, что если представить, что эти нейроны (фрагменты воспоминаний) – вершины графа, а связи (ассоциации) – это рёбра, то получается дерево. Врачи называют его деревом твёрдых ассоциаций.

Чтобы окончательно свести нашего больного с ума (временно, в чём и заключается метод врачей) достаточно сделать так, чтобы не существовало фрагмента, из которого по ассоциациям можно пройтись **цепочкой** более чем по k другим фрагментам, не "заходя" в один и тот же фрагмент более одного раза.

Как только эта цель достигается, оставшиеся нейроны с фрагментами воспоминаний отмирают, человек окончательно теряет связь с реальностью и начинается процесс восстановления памяти, которая, как показала практика, становится лучше прежнего. При восстановлении верно определяется, является воспоминание реальным или оно является плодом фантазии больного, а граница разделения реального и иллюзорного становится чётче. Именно поэтому этот метод рекомендуется людям с болезнями, вызванными неправильной работой памяти и мышления.

Для того, чтобы разрушить память до конца, у врачей есть специальный вирус-киллер, один экземпляр которого может добраться до любого, но только до одного заранее заданного нейрона и избавиться от него и связей, соединяющих его с другими нейронами. Однако врачи не разбираются в теории графов и тратят много вирусов-киллеров почём зря.

Помогите врачам минимизировать количество экземпляров вируса-киллера, которое они потратят на больных.

Формат входных данных

В каждом тесте описывается состояние памяти после применения нейровируса ровно одного больного.

В первой строке находятся числа n и k ($1 \leqslant n, k \leqslant 100\,000$) — число фрагментов воспоминаний и число k, описание которого ищите на предыдущей странице. Сами фрагменты воспоминаний пронумерованы от 1 до n.

Каждая из следующих $n - 1$ строк содержит по два целых числа u и v ($1 \leqslant u, v \leqslant n, u \neq v$), говорящих о существовании ассоциации между двумя фрагментами.

Гарантируется, что эти данные образуют дерево твёрдых ассоциаций.

Формат выходных данных

В первой строке выведите минимальное количество экземпляров вируса-киллера, которое нужно потратить на больного.

Во второй строке выведите номера нейронов, к которым нужно послать вирусов-киллеров.

Если подходящих списков номеров нейронов несколько – выведите любой.

Примеры

Пример №1

Стандартный ввод
4 2 1 2 2 3 3 4

Стандартный вывод
1 3

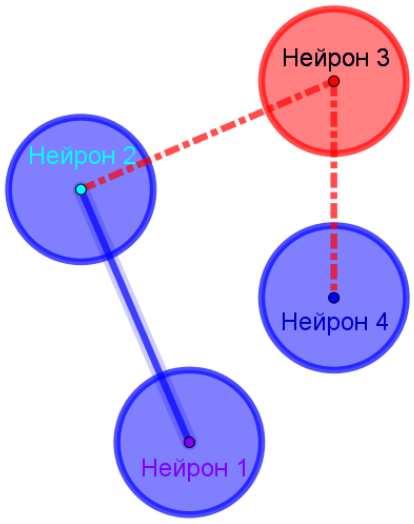


Рис. III.1.4: Рисунок к примеру 1

Пример №2

Стандартный ввод
6 3 1 2 2 3 2 4 2 5 4 6

Стандартный вывод
1 6

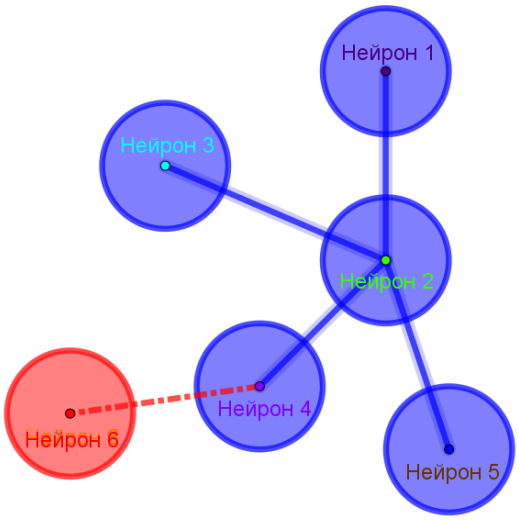


Рис. III.1.5: Рисунок к примеру 2

Пример №3

Стандартный ввод
6 2 1 2 2 3 2 4 2 5 4 6

Стандартный вывод
1 2

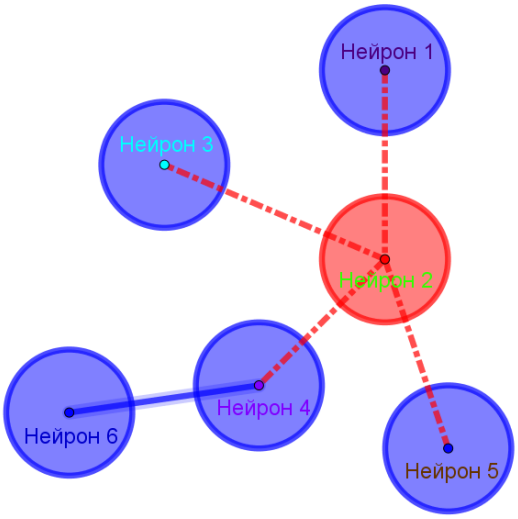


Рис. III.1.6: Рисунок к примеру 3

Пояснение к примерам

Ответом в первом примере мог быть и второй нейрон, но не первый или четвёртый, так как в этих случаях остаётся цепочка из трёх нейронов.

Ответ в третьем примере также является ответом и для второго, но ответ второго теста не является ответом для третьего, так как при отмирании 6-го нейрона в оставшемся дереве есть цепочка из трёх нейронов (например, $3 \leftrightarrow 2 \leftrightarrow 5$).

Способ оценки работы

Подзадача	Баллы	Дополнительные ограничения	Необходимые подзадачи
0	0	Тесты из условия	–
1	20	$n \leqslant 15$	0
2	15	Каждый нейрон имеет не более двух связей	–
3	15	$k = 1$	–
4	15	$k = 2$	3
5	10	$n \leqslant 100$	0, 1
6	10	$n \leqslant 1000$	0, 1, 5
7	15	Нет дополнительных ограничений	0, 1, 2, 3, 4, 5, 6

Баллы за подзадачу будут начислены в случае прохождения всех тестов необходимых подзадач и всех тестов самой подзадачи.

Для проверки результата использовался следующий код на языке C++

```cpp
#include "testlib.h"

#include <bits/stdc++.h>

using namespace std;

int main(int argc, char * argv[]) {
  registerTestlibCmd(argc, argv);

  int n = inf.readInt(1, 100000, "n");
  inf.readSpace();
  int k = inf.readInt(1, 100000, "k");
  inf.readEoln();
  vector < vector < int > > edges(n);
  for (int i = 0; i < n - 1; i++) {
    int u = inf.readInt(1, n, "u") - 1;
    inf.readSpace();
    int v = inf.readInt(1, n, "v") - 1;
    inf.readEoln();
    edges[u].push_back(v);
    edges[v].push_back(u);
  }
  inf.readEof();

  long long ja = ans.readLong();
  ans.readEoln();
  vector < bool > used(n, 0);
  for (int i = 0; i < ja; i++) {
    int v = ans.readInt(1, n, "next v") - 1;
    if (used[v])
      quitf(_fail, "wrong sequence (double %d)", v + 1);
    used[v] = 1;
  }

  vector < int > cnt(n, 0), len(n, 0);
  queue < int > q;

  for (int i = 0; i < n; i++) {
    if (used[i]) continue;
    for (int j: edges[i]) {
      if (!used[j]) {
        cnt[i]++;
      }
    }
  }
  for (int i = 0; i < n; i++) {
    if (!used[i] && cnt[i] <= 1) {
      used[i] = true;
      q.push(i);
    }
  }

  int answer = 0;
  while (!q.empty()) {
    int v = q.front();
    q.pop();
    int max1 = 0, max2 = 0;
    for (int u: edges[v]) {
      if (len[u] > 0) {
        if (len[u] > max1) {
```

```
          max2 = max1;
          max1 = len[u];
        } else if (len[u] > max2) {
          max2 = len[u];
        }
      } else if (!used[u] && --cnt[u] == 1) {
        used[u] = true;
        q.push(u);
      }
    }
    answer = max(answer, max1 + 1 + max2);
    len[v] = max1 + 1;
  }

  if (answer > k)
    quitf(_fail, "wrong sequense (%d > %d)", answer, k);

  long long pa = ouf.readLong();
  ouf.readEoln();
  if (ja < pa)
    quitf(_wa, "wrong answer");
  fill(used.begin(), used.end(), 0);
  for (int i = 0; i < pa; i++) {
    int v = ouf.readInt(1, n, "next v") - 1;
    if (used[v])
      quitf(_wa, "wrong sequence (double %d)", v + 1);
    used[v] = 1;
  }

  fill(cnt.begin(), cnt.end(), 0);
  fill(len.begin(), len.end(), 0);

  for (int i = 0; i < n; i++) {
    if (used[i]) continue;
    for (int j: edges[i]) {
      if (!used[j]) {
        cnt[i]++;
      }
    }
  }
  for (int i = 0; i < n; i++) {
    if (!used[i] && cnt[i] <= 1) {
      used[i] = true;
      q.push(i);
    }
  }

  answer = 0;
  while (!q.empty()) {
    int v = q.front();
    q.pop();
    int max1 = 0, max2 = 0;
    for (int u: edges[v]) {
      if (len[u] > 0) {
        if (len[u] > max1) {
          max2 = max1;
          max1 = len[u];
        } else if (len[u] > max2) {
          max2 = len[u];
        }
```

```
121         } else if (!used[u] && --cnt[u] == 1) {
122           used[u] = true;
123           q.push(u);
124         }
125       }
126       answer = max(answer, max1 + 1 + max2);
127       len[v] = max1 + 1;
128     }
129
130     if (answer > k)
131       quitf(_wa, "wrong sequense (%d > %d)", answer, k);
132
133     if (ja > pa)
134       quitf(_fail, "participant has better answer");
135
136     quitf(_ok, "%d numbers", n);
137
138 }
```

Решение

Подзадача 1

Переберём все возможные множества удалённых вершин и для каждого проверим, есть ли в полученном лесе (то есть графе, каждая компонента связности которого является деревом) путь длины более чем k

Подзадача 2

Когда с каждой вершиной дерева связано не более двух других вершин, дерево превращается в «цепочку».

Одним из решений для такого графа является удаление каждой $k+1$-й вершины на пути от одного из листьев.

Иллюстрация для $k = 2$:

Подзадача 3

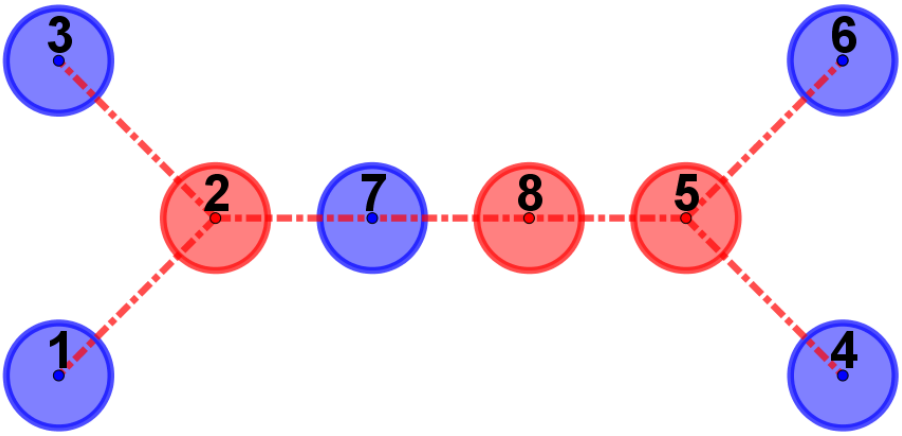
При $k = 1$ в итоговом графе каждая компонента связности будет состоять из одной вершины.

Таким образом, задача сводится к задаче о независимом множестве вершин дерева.

Для решения этой задачи можно применить жадный алгоритм, который кратко можно изложить следующим образом:

Пока в графе есть вершины, будем по одному удалять листья и связанные с ними вершины.

Вершины, которые удалялись не из-за того, что являются листьями, являются одним из оптимальных ответов для нашей задачи.



Вершины пронумерованы в порядке удаления.

Полное решение

Задачу в терминах теории графов можно сформулировать так:

Дано дерево из n вершин и число k. Нужно удалить как можно меньше вершин, чтобы в оставшемся графе не существовало пути длиной больше чем k.

Давайте решим задачу для вершины v и ее поддерева.

Для начала нужно выполнить условие для поддеревьев детей вершины v, то есть в поддеревьях детей вершины v нет пути длиной больше чем k.

Теперь предположим, что мы нашли ответ такой, что удалено минимальное количество вершин и последняя удаленная вершина выбрана так, чтобы минимизировать длину остаточного пути из соответствующего поддерева.

Теперь, когда мы решили задачу для поддеревьев детей вершины v, у нас есть какие-то остаточные пути из этих поддеревьев. Пусть два самых длинных из них имеют длины mx_1 и mx_2 соответственно (если какого-то пути не существует, то будем считать, что его длина равна нулю). Если мы оставим вершину v в графе, то останется и путь длиной $mx_1 + mx_2 + 1$. Если $mx_1 + mx_2 + 1 > k$, то мы обязаны удалить вершину v; иначе – можем ее оставить. Если мы удаляем вершину v, то длина остаточного пути, идущего из вершины v в её поддерево, будет считаться равной нулю. Иначе она равна $mx_1 + 1$.

Запустив данное решение для любой вершины в качестве корня дерева, можно решить задачу для всего дерева (удобно использовать обход в глубину).

Докажем оптимальность такого решения методом математической индукции.

Рассмотрим поддерево из одной вершины. Очевидно, что алгоритм отработает оптимально: ничего не удалит и оставит минимальный по длине путь длины 1.

Рассмотрим поддерево вершины v. Для каждого поддерева детей вершины v мы решили задачу оптимально нашим алгоритмом. Докажем, что наш алгоритм найдёт оптимальное решение для всего поддерева вершины v.

Если длина самого длинного пути в поддереве вершины v превышает k, то мы обязаны удалить как минимум одну вершину. Наш алгоритм удаляет ровно одну вершину и это вершина v.

Предположим, что мы можем удалить какую-то вершину в поддереве вершины v так, чтобы длина самого длинного пути в поддереве вершины v не превышала k, и это не вершина v. Если мы удалим её вместо вершины v, то останется еще какой-то ненулевой остаточный путь из вершины v в её поддерево. А при равенстве количества удаленных вершин мы должны минимизировать длину остаточного пути. Отказ от минимизации длины остаточного пути, очевидно, может привести к нахождение неоптимального ответа.

Если предположить, что мы можем не удалять ещё одну вершину, а заменить удаление одной из включённых в ответ вершин на удаление вершины v, то мы приходим к противоречию: если мы можем отменить удаление одной из вершин в поддереве одного из детей вершины v, то найденный нами ответ для поддерева этого ребёнка вершины v неоптимален, так как мы можем уменьшить количество удалённых вершин в поддереве этого ребёнка вершины v.

Если длина самого длинного пути в поддереве вершины v не превышает k, то мы ничего не удаляем. Количество удаленных вершин оптимально. Несложно доказать, что длина остаточного пути вершины v тоже будет оптимальной (она равна $mx_1 + 1$ и так как длина пути mx_1 была оптимальной, то оставшийся путь тоже оптимален).

Пример программы-решения

Ниже представлено решение на языке Python 3

```
n, k = map(int, input().split())
graph = [[] for i in range(n)]
for i in range(n - 1):
    v, u = map(int, input().split())
    v -= 1
    u -= 1
    graph[v].append(u)
    graph[u].append(v)

deleted = [False] * n
used = [False] * n
cnt = [len(graph[i]) for i in range(n)]
len = [0] * n
q = [0] * n
ql, qr = 0, 0

for i in range(n):
    if cnt[i] < 2:
        used[i] = True
        q[qr] = i
        qr += 1

answer = 0
while ql < qr:
    v = q[ql]
    ql += 1
    max1, max2 = 0, 0
    for u in graph[v]:
        if len[u] > 0:
            if len[u] > max1:
                max1, max2 = len[u], max1
            elif len[u] > max2:
                max2 = len[u]
```

```
        elif not used[u]:
            cnt[u] -= 1
            if cnt[u] == 1:
                used[u] = True
                q[qr] = u
                qr += 1

    if max1 + 1 + max2 > k:
        answer += 1
        deleted[v] = True
    else:
        len[v] = max1 + 1

print(answer)
for i in range(n):
    if deleted[i]:
        print(i + 1, end=' ')
```